

---

# **cepton\_sdk Documentation**

**Cepton Technologies**

**Oct 31, 2019**



# CONTENTS

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Errors</b>	<b>3</b>
<b>3</b>	<b>Setup</b>	<b>5</b>
<b>4</b>	<b>General</b>	<b>7</b>
<b>5</b>	<b>Sensors</b>	<b>9</b>
<b>6</b>	<b>Points</b>	<b>11</b>
<b>7</b>	<b>Serial</b>	<b>13</b>
<b>8</b>	<b>Capture Replay</b>	<b>15</b>
<b>9</b>	<b>Export</b>	<b>17</b>
<b>10</b>	<b>Samples</b>	<b>19</b>



## **OVERVIEW**

If a method is undocumented, consult the *C/C++ SDK* documentation, since many methods in this library are just wrapper functions.

### **1.1 Timestamps**

Unless otherwise marked, all timestamps are seconds since the Unix epoch (UTC). Note that this differs from the *C/C++* interface which uses microseconds.



---

CHAPTER  
TWO

---

ERRORS





### **3.1 Types**

### **3.2 Methods**



**GENERAL**

API for code that is agnostic to live/replay mode.



## **SENSORS**

### **5.1 Types**

### **5.2 Methods**



## 6.1 Types

All point array classes support numpy indexing and assignment as if they were 1-d arrays:

```
1 n_points = len(points_1)
2 points_2[10:20] = points_1[:10]
```

Multiple point arrays can also be combined:

```
1 points = cepton_sdk.combine_points([points_1, points_2])
```

## 6.2 Methods

See *Listen*.

The following methods return points directly from the C SDK callback.

There are also listener classes that seamlessly handle accumulation and waiting.

## 6.3 Export





## 7.1 Methods

The following methods return serial lines directly from the C SDK callback.

There is also a listener class that seamlessly handle accumulation.



## CAPTURE REPLAY

To open/close capture files, use `cepton_sdk.initialize` and `cepton_sdk.deinitialize` methods respectively. The high level API methods will automatically resume the capture replay as necessary.



## **EXPORT**

Methods to import/export points to common file formats.



## 10.1 Multiple Sensors

Listing 1: samples/multiple\_sensors.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for getting points from multiple sensors simultaneously.
4  """
5
6  import pprint
7
8  import cepton_sdk
9  import cepton_sdk.plot
10 from common import *
11
12 if __name__ == "__main__":
13     # Variables
14     capture_path = get_sample_capture_path()
15
16     # Initialize
17     cepton_sdk.initialize(capture_path=capture_path, enable_wait=True)
18
19     # Get sensors
20     sensors_dict = cepton_sdk.get_sensors()
21
22     # Get points
23     listener = cepton_sdk.FramesListener()
24     points_dict = listener.get_points()
25     del listener
26     points_list = next(iter(points_dict.values()))
27     points = points_list[0]
28
29     # Plot
30     cepton_sdk.plot.plot_points(points)
```

## 10.2 Single Sensor

Listing 2: samples/single\_sensor.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for getting points from a single sensor.
4  """
5
6  import pprint
7
8  import numpy
9
10 import cepton_sdk
11 import cepton_sdk.plot
12 from common import *
13
14 if __name__ == "__main__":
15     # Variables
16     capture_path = get_sample_capture_path()
17
18     # Initialize
19     cepton_sdk.initialize(capture_path=capture_path, enable_wait=True)
20
21     # Get sensor
22     sensor = cepton_sdk.Sensor.create_by_index(0)
23     pprint.pprint(sensor.information.to_dict())
24
25     # Get points
26     listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
27     points_list = listener.get_points()
28     del listener
29     points = points_list[0]
30
31     # Plot
32     cepton_sdk.plot.plot_points(points)
```

## 10.3 Advanced

### 10.3.1 Listen

Listing 3: samples/advanced/listen.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for the different methods of getting points.
4  """
5
6  import numpy
7
8  import cepton_sdk
9  from common import *
10
11
12 def on_frame(serial_number, points):
13     print("Received {} points from sensor {}".format(
```

(continues on next page)



(continued from previous page)

```
14         len(points), serial_number))
15
16
17 if __name__ == "__main__":
18     # Initialize
19     cepton_sdk.initialize(capture_path=get_sample_capture_path())
20     sensors_dict = cepton_sdk.get_sensors()
21     sensor = next(iter(sensors_dict.values()))
22
23     callback_id = cepton_sdk.listen_frames(on_frame)
24     cepton_sdk.wait(0.1)
25     cepton_sdk.unlisten_frames(callback_id)
26
27     # Get next frames for all sensors. Wait until data is available.
28     listener = cepton_sdk.FramesListener()
29     points_dict = listener.get_points()
30     del listener
31
32     # Get next frames for single sensor. Wait until data is available.
33     listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
34     points_list = listener.get_points()
35     del listener
36
37     # Get large chunk of data
38     listener = cepton_sdk.FramesListener()
39     cepton_sdk.wait(10)
40     points_dict = listener.get_points()
41     del listener
42     points = cepton_sdk.combine_points(points_dict[sensor.serial_number])
43     print("Received {} seconds of data from sensor {}".format(
44         numpy.ptp(points.timestamps), sensor.serial_number))
```